```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay, precision_score, recall_score, accuracy_score

data = pd.read_csv("/content/emails.csv")
data
```

```
{"type":"dataframe","variable_name":"data"}
```

```
data = data.drop('Email No.', axis=1)

data.shape
```

```
(5172, 3001)
```

```
data.describe()
```

```
{"type":"dataframe"}
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3001 entries, the to Prediction
dtypes: int64(3001)
memory usage: 118.4 MB
```

```
data['Prediction'].value_counts()
```

```
Prediction
0    3672
1    1500
Name: count, dtype: int64
```

```
X = data.drop('Prediction', axis = 1)
y = data['Prediction']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.20)

from sklearn.neighbors import KNeighborsClassifier
from sklearn.impute import SimpleImputer

# Impute missing values
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)
```

```
neigh = KNeighborsClassifier(n_neighbors = 2)
neigh.fit(X_train_imputed, y_train)
```

```
KNeighborsClassifier(n_neighbors=2)
```

```
y_pred = neigh.predict(X_test)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/utils/
validation.py:2732: UserWarning: X has feature names, but
KNeighborsClassifier was fitted without feature names
  warnings.warn(
```

```
neigh.score(X_train, y_train)
neigh.score(X_test, y_test)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/utils/
validation.py:2732: UserWarning: X has feature names, but
KNeighborsClassifier was fitted without feature names
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:27
32: UserWarning: X has feature names, but KNeighborsClassifier was
fitted without feature names
  warnings.warn(
```

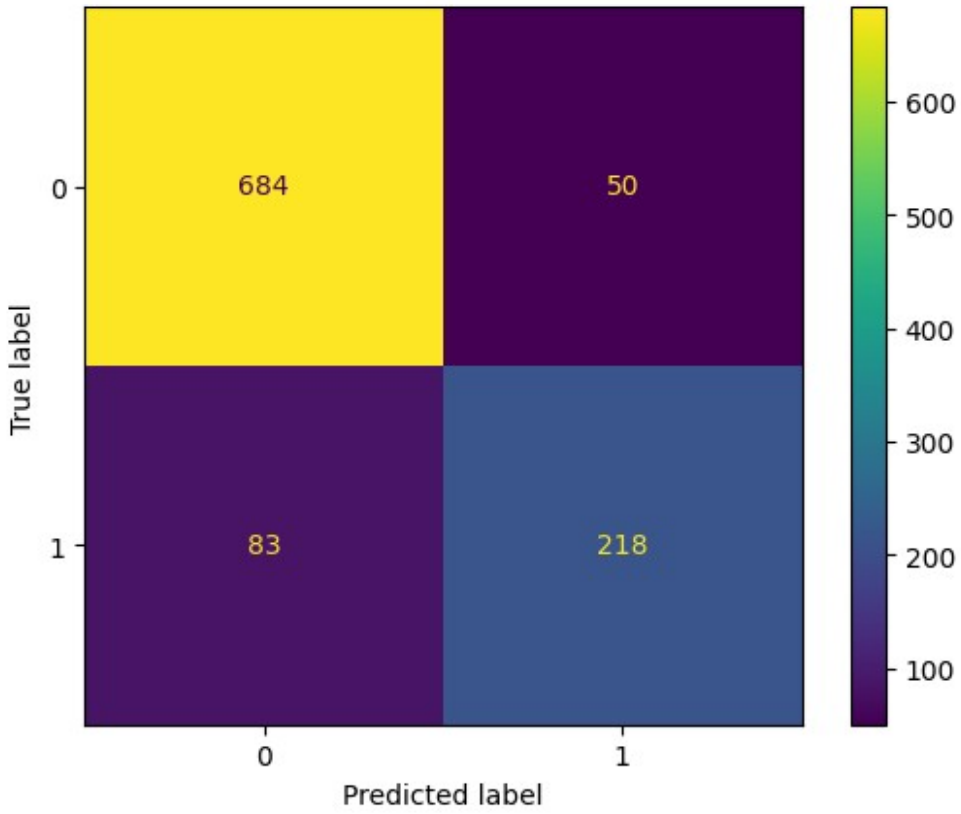
```
0.8714975845410629
```

```
print("Confusion Matrix: ")
cm = confusion_matrix(y_test, y_pred)
cm
```

```
Confusion Matrix:
```

```
array([[684,  50],
       [ 83, 218]])
```

```
mat = ConfusionMatrixDisplay(confusion_matrix = cm)
mat.plot()
plt.show()
```




```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.93      0.91       734
           1       0.81      0.72      0.77       301

    accuracy                           0.87      1035
   macro avg       0.85      0.83      0.84      1035
weighted avg       0.87      0.87      0.87      1035
```

```
print("accuracy_score: ")
accuracy_score(y_test, y_pred)
```

```
accuracy_score:
```

```
0.8714975845410629
```

```
print("precision_score: ")
precision_score(y_test, y_pred)
```

```
precision_score:
```

```
0.8134328358208955
```

```
print("recall_score: ")
recall_score(y_test, y_pred)
```

```
recall_score:
```

```
0.7242524916943521
```

```
print("Error: ")
1-accuracy_score(y_test, y_pred)
```

```
Error:
```

```
0.12850241545893715
```

```
from sklearn.svm import SVC
SVM = SVC(gamma = 'auto')
SVM.fit(X_train, y_train)
```

```
SVC(gamma='auto')
```

```
y_pred = SVM.predict(X_test)
```

```
SVM.score(X_train, y_train)
SVM.score(X_test, y_test)
```

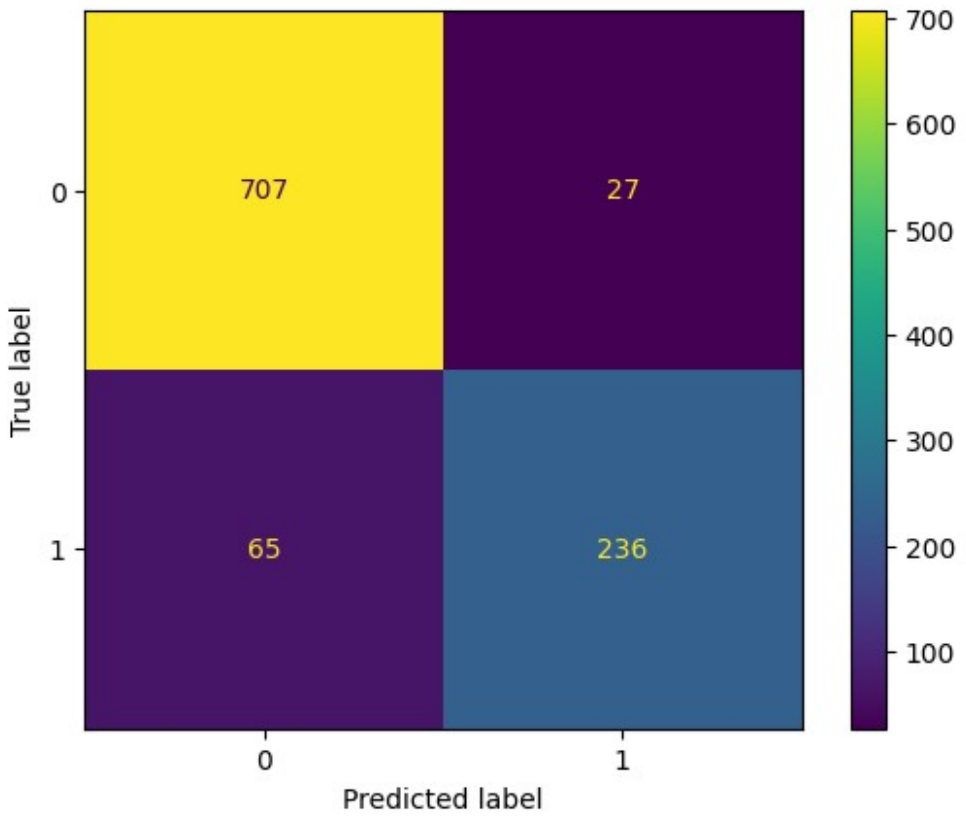
```
0.9468599033816425
```

```
print("Confusion Matrix: ")
cm = confusion_matrix(y_test, y_pred)
cm
```

```
Confusion Matrix:
```

```
array([[707,  27],
       [ 65, 236]])
```

```
mat = ConfusionMatrixDisplay(confusion_matrix = cm)
mat.plot()
plt.show()
```




```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       734
           1       0.90      0.78      0.84       301

    accuracy                           0.91      1035
   macro avg       0.91      0.87      0.89      1035
weighted avg       0.91      0.91      0.91      1035
```